  如果你基于Qt SDK 5.3.1来创建一个Qt Quick App项目，项目模板为你准备的main.qml文档的根元素是ApplicationWindow或Window。这次我们就以ApplicationWindow为例，围绕着它实现一个综合实例：文件查看器。通过文件查看器的实现，我们来再次领略一下Qt Quick的犀利。

    版权所有foruok，转载请注明出处：<http://blog.csdn.net/foruok>。

    本实例将会用到下列特性：

* ApplicationWindow
* MenuBar
* ToolBar、ToolButton
* Action
* StatusBar
* MediaPlayer
* Image
* XMLHttpRequest
* ColorDialog
* FileDialog
* TextArea
* 动态创建QML组件
* 多界面切换

    我们之前的文章都是就某一个主题来展开的，这次要出个大招。先看点儿图。

**文件查看器效果**

    图1，初始状态

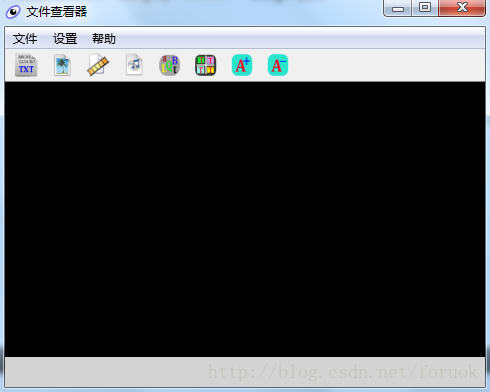


                  图1 初始状态

    图2 是浏览文本文件的效果：

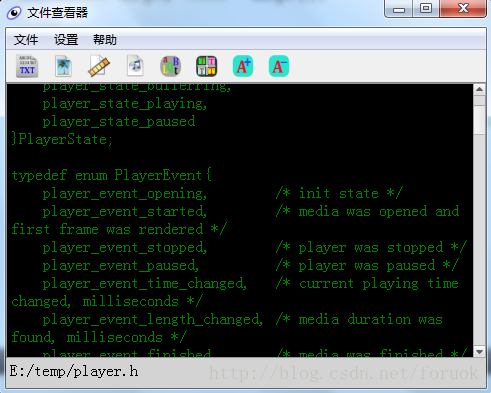


              图2 浏览文本文件

    图3是浏览图片：

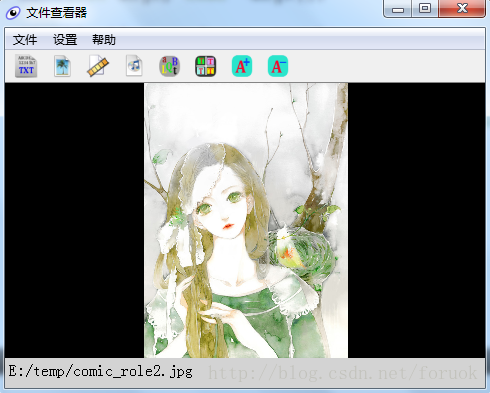


           图3 浏览图片

    图4是播放视频：



              图4 播放视频

    图5是播放音乐：

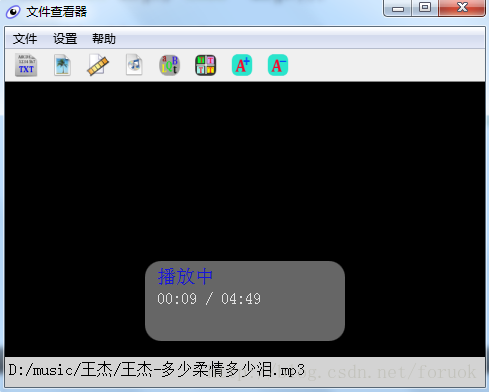


               图5 播放音乐

    好啦，看代码前，先大概介绍下用到的Qt Quick元素。

    Image、动态创建组件、界面切换这些没什么好说的，我之前的文章中已经涉及到。只讲新的了。

**ApplicationWindow**

    ApplicationWindow，类似Qt C++中的QMainWindow，请对照着理解。

    ApplicationWindow有菜单栏(menuBar属性）、工具栏（toolBar属性）、状态栏（statusBar属性）。咦，没有centralWidget哈，别急，有的，就是ContentItem，就是那个你不与任何属性绑定的Item。例如：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ApplicationWindow{
2. ...
3. Rectangle{
4. id: centralView;
5. anchors.fill: parent;
6. color: "red";
7. }
8. }

ApplicationWindow{

...

Rectangle{

id: centralView;

anchors.fill: parent;

color: "red";

}

}

    上面代码中的centralView就对应QMainWindow的centralWidget了。

    看起来没有DockWidgets……兄弟，别要求太高了，你想要么，真的想要么，你真的确定你想要么？好吧，自己实现就好了，QML里很多元素的哈。

**MenuBar**

    MenuBar就是菜单栏一长条区域了，它干两件事：

1. 维护一个Menu列表（menus属性）
2. 绘制菜单栏背景色（style属性）

**Menu**

    看截图中的“文件”、“设置”、“帮助”三个菜单，点一下弹出一个子菜单，Menu就代表文件这个菜单和它下面的子菜单。

    列表属性items指向Menu的子菜单项，它的类型是list<Object>，是默认属性。

    title属性是string类型，你看到的“文件”、“设置”等字样就是它指定的。

    有这两个属性，Menu就可以开始干活了。看Qt帮助里的示例代码片段：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. Menu {
2. title: "Edit"
4. MenuItem {
5. text: "Cut"
6. shortcut: "Ctrl+X"
7. onTriggered: ...
8. }
10. MenuItem {
11. text: "Copy"
12. shortcut: "Ctrl+C"
13. onTriggered: ...
14. }
16. MenuItem {
17. text: "Paste"
18. shortcut: "Ctrl+V"
19. onTriggered: ...
20. }
22. MenuSeparator { }
24. Menu {
25. title: "More Stuff"
27. MenuItem {
28. text: "Do Nothing"
29. }
30. }
31. }

Menu {

title: "Edit"

MenuItem {

text: "Cut"

shortcut: "Ctrl+X"

onTriggered: ...

}

MenuItem {

text: "Copy"

shortcut: "Ctrl+C"

onTriggered: ...

}

MenuItem {

text: "Paste"

shortcut: "Ctrl+V"

onTriggered: ...

}

MenuSeparator { }

Menu {

title: "More Stuff"

MenuItem {

text: "Do Nothing"

}

}

}

**MenuItem**

    Menu的孩子啊，最受待见的就是MenuItem了。

    MenuItem代表一个具体的菜单项，点一下就干一件事情的角色。你可以实现onTriggered信号处理响应用户对它的选择。

    MenuItem的text属性指定菜单文字，iconSource属性指定菜单图标。

    Action属性很强大哦，MenuItem的text、iconSource、trigger信号等实现的效果，都可以通过Action来实现，这两种方式是等同的。咱们来看Action。

**Action**

    Action类有text、iconSource等属性，还有toggled、triggered两个信号。这些在MenuItem里有对应的属性，不必多说了。

    使用Action的一大好处是，你可以给它指定一个id（比如叫open），然后在使用ToolButton构造ToolBar时指定ToolBatton的action属性为之前定义的id为open的那个action，这样工具栏的按钮就和菜单关联起来了。

    好啦，总结一下，实现菜单栏的典型代码结构是酱紫的：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. MenuBar {
2. Menu{
3. title:"文件";
4. MenuItem{
5. text:"打开";
6. iconSource: "res/ic\_open.png";
7. onTriggered:{
8. ...
9. }
11. }
12. MenuItem{
13. action: Action {
14. text:"保存";
15. iconSource: "res/ic\_save.png";
16. onTriggered:{
17. ...
18. }
19. }
20. }
21. }
22. }

MenuBar {

Menu{

title:"文件";

MenuItem{

text:"打开";

iconSource: "res/ic\_open.png";

onTriggered:{

...

}

}

MenuItem{

action: Action {

text:"保存";

iconSource: "res/ic\_save.png";

onTriggered:{

...

}

}

}

}

}

    如你所见，我使用了两种构造MenuItem的方式，一种用到了Action，一种没有。

**ToolBar**

    ToolBar就是工具栏对应的类，它只有一个属性，contentItem，类型为Item。一般我们可以将一个Row或者RowLayout对象赋值给contentItem，而Row或RowLayout则管理一组ToolButton来作为工具栏上的按钮。

**ToolButton**

    ToolButton是Button的派生类，专为ToolBar而生，一般情况下定义ToolButton对象时只需要指定其iconSource属性即可。例如：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ToolButton {
2. iconSource: "res/ic\_open.png";
3. }

ToolButton {

iconSource: "res/ic\_open.png";

}

    还有一种方式是将一个已定义好的Action对象关联到ToolButton对象上。例如：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ToolButton{
2. action: openAction;
3. }

ToolButton{

action: openAction;

}

    这样的话，ToolButton会使用Action定义的iconSource或iconName作为其图标。

    好啦，构造工具栏的典型代码结构如下：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ToolBar{
2. RowLayout {
3. ToolButton{
4. action: textAction;
5. }
6. ToolButton{
7. action: imageAction;
8. }
9. ToolButton{
10. iconSource: "res/exit.png";
11. onClicked:{
12. ...
13. }
14. }
15. }
16. }

ToolBar{

RowLayout {

ToolButton{

action: textAction;

}

ToolButton{

action: imageAction;

}

ToolButton{

iconSource: "res/exit.png";

onClicked:{

...

}

}

}

}

**状态栏**

    ApplicationWindow的属性statusBar代表状态栏，其类型为Item，你可以将任意的Item赋值给它，可以随心所欲构建你妖娆多姿的状态栏。比如这样：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ApplicationWindow{
2. statusBar: Text {
3. text: "status bar";
4. color: "blue";
5. }
6. }

ApplicationWindow{

statusBar: Text {

text: "status bar";

color: "blue";

}

}

**MediaPlayer**

    Qt Quick里，播放视频、音频文件都直接使用MediaPlayer类，它是万能的，不要说你万万没想到哈。

    对于音乐，最简单，设置source属性，调用play()方法，就可以了。例如：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ApplicationWindow{
2. ...
3. MediaPlayer{
4. id: player;
5. source: "xxx.mp3";
6. }
7. Component.onCompleted:{
8. player.play();
9. }
10. }

ApplicationWindow{

...

MediaPlayer{

id: player;

source: "xxx.mp3";

}

Component.onCompleted:{

player.play();

}

}

    对于视频，MediaPlayer还得寻求场外帮助，求助对象就是它的好基友：VideoOutput ！简单的示例代码：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. ApplicationWindow{
2. ...
3. MediaPlayer {
4. id: player;
5. source: "test.ts";
6. }
8. VideoOutput {
9. anchors.fill: parent;
10. source: player;
11. }
13. Component.onCompleted: player.play();
14. }

ApplicationWindow{

...

MediaPlayer {

id: player;

source: "test.ts";

}

VideoOutput {

anchors.fill: parent;

source: player;

}

Component.onCompleted: player.play();

}

    哦，黄小琥唱过的歌：没那么简单。是的，实际做项目比较复杂，你还要关注各种播放状态，要seek，要pause，要处理错误，简直烦死人了。请移步到Qt帮助里了解详情。

**XMLHttpRequest**

    在Qt Quick里，要访问网络肿么办泥？答案是：XMLHttpRequest ！

    不要被它傲娇的外表迷惑，以为它只接受XML文档，其实，它什么都能处理，txt、html、json、binary……它温柔坚定强悍无比。

    本实例用它来加载本地的文本文件，耶，这样都可以哈。谁叫Qt Quick不提供直接访问本地文件的类库呢！我可不想跑到C++里用QFile、QTextStream这对黄金搭档。

    XMLHttpRequest的文档，Qt帮助里语焉不详，只有一个示例，请看这里：

<http://www.w3school.com.cn/xml/xml_http.asp>

**TextArea**

    这有什么好讲的，看Qt帮助吧。只提一点：

    TextArea自动处理翻页按键、上下键、鼠标中键，正确的滚动文本；而TextEdit，抱歉，我是来打酱油的。

**标准对话框**

    Qt Quick提供了很多标准对话框，比如FileDialog用来选择文件或文件夹，ColorDialog用来选择颜色，MessageDialog用来显示一些提示信息。这些我们实例中用到了，参考Qt帮助吧。

    我只说点儿经验。

    我一开始使用qmlscene来加载main.qml，出来的界面比较正常，工具栏的图标、菜单项前也有图标。可是当我创建了一个Qt Quick App，灵异事件发生了：

    菜单项前面没有图标了……

    工具栏图标好大好大……

    颜色对话框、消息框，点击右上角的关闭按钮，收不到rejected信号啊……

    查了老半天，猛回头，警世钟响起，我了悟了。原来是酱紫的：

    Qt Quick提供了这些标准对话框的默认实现，如果应用运行的平台没有可用的，就用这些默认实现。那在Windows上，如果你的main()函数，使用QGuiApplication而非QApplication，就会用到Qt Quick实现的版本，一切都变了模样

**资源管理**

    Qt SDK 5.3之后，Qt Creator创建的Qt Quick App项目，就为我们建立了一个qrc文件，把main.qml扔里啦。我在实例中也把很多图标扔里了。

    使用qrc来管理资源，这是跨平台的，推荐使用。

    我还自己画了些图标，真费劲，弄得也不好看。不过应用的大眼睛图标，看起来还像那么一回事儿。

**源代码**

    前戏太长，也许你已经失去了兴趣。好吧，G点来咧。

**QML代码**

    版权所有foruok，转载请注明出处：<http://blog.csdn.net/foruok>。

    所有QML代码都在这里了，竟然有450行啊亲。

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. import QtQuick 2.2
2. import QtQuick.Window 2.1
3. import QtQuick.Controls 1.2
4. import QtQuick.Controls.Styles 1.2
5. import QtQuick.Layouts 1.1
6. import QtQuick.Dialogs 1.1
7. import QtMultimedia 5.0
9. ApplicationWindow {
10. visible: true
11. width: 480
12. height: 360;
13. color: "black";
14. title: "文件查看器";
15. id: root;
16. property var aboutDlg: null;
17. property var colorDlg: null;
18. property color textColor: "green";
19. property color textBackgroundColor: "black";
21. menuBar: MenuBar{
22. Menu {
23. title: "文件";
24. MenuItem{
25. iconSource: "res/txtFile.png";
26. action: Action{
27. id: textAction;
28. iconSource: "res/txtFile.png";
29. text: "文本文件";
30. onTriggered: {
31. fileDialog.selectedNameFilter = fileDialog.nameFilters[0];
32. fileDialog.open();
33. }
34. tooltip: "打开txt等文本文件";
35. }
36. }
37. MenuItem{
38. action: Action {
39. id: imageAction;
40. text: "图片";
41. iconSource: "res/imageFile.png";
42. onTriggered: {
43. fileDialog.selectedNameFilter = fileDialog.nameFilters[1];
44. fileDialog.open();
45. }
46. tooltip: "打开jpg等格式的图片";
47. }
48. }
49. MenuItem{
50. action: Action {
51. id: videoAction;
52. iconSource: "res/videoFile.png";
53. text: "视频";
54. onTriggered: {
55. fileDialog.selectedNameFilter = fileDialog.nameFilters[2];
56. fileDialog.open();
57. }
58. tooltip: "打开TS、MKV、MP4等格式的文件";
59. }
60. }
61. MenuItem{
62. action: Action {
63. id: audioAction;
64. iconSource: "res/audioFile.png";
65. text: "音乐";
66. onTriggered: {
67. fileDialog.selectedNameFilter = fileDialog.nameFilters[3];
68. fileDialog.open();
69. }
70. tooltip: "打开mp3、wma等格式的文件";
71. }
72. }
73. MenuItem{
74. text: "退出";
75. onTriggered: Qt.quit();
76. }
77. }
78. Menu {
79. title: "设置";
80. MenuItem {
81. action: Action {
82. id: textColorAction;
83. iconSource: "res/ic\_textcolor.png";
84. text: "文字颜色";
85. onTriggered: root.selectColor(root.onTextColorSelected);
86. }
87. }
88. MenuItem {
89. action: Action{
90. id: backgroundColorAction;
91. iconSource: "res/ic\_bkgndcolor.png";
92. text: "文字背景色";
93. onTriggered: root.selectColor(root.onTextBackgroundColorSelected);
94. }
95. }
96. MenuItem {
97. action: Action{
98. id: fontSizeAddAction;
99. iconSource: "res/ic\_fontsize2.png";
100. text: "增大字体";
101. onTriggered: textView.font.pointSize += 1;
102. }
103. }
104. MenuItem {
105. action: Action{
106. id: fontSizeMinusAction;
107. iconSource: "res/ic\_fontsize1.png";
108. text: "减小字体";
109. onTriggered: textView.font.pointSize -= 1;
110. }
111. }
112. }
113. Menu {
114. title: "帮助";
115. MenuItem{
116. text: "关于";
117. onTriggered: root.showAbout();
118. }
119. MenuItem{
120. text: "访问作者博客";
121. onTriggered: Qt.openUrlExternally("http://blog.csdn.net/foruok");
122. }
123. }
124. }
126. toolBar: ToolBar{
127. RowLayout {
128. ToolButton{
129. action: textAction;
130. }
131. ToolButton{
132. action: imageAction;
133. }
134. ToolButton{
135. action: videoAction;
136. }
137. ToolButton{
138. action: audioAction;
139. }
140. ToolButton{
141. action: textColorAction;
142. }
143. ToolButton {
144. action: backgroundColorAction;
145. }
146. ToolButton {
147. action: fontSizeAddAction;
148. }
149. ToolButton {
150. action: fontSizeMinusAction;
151. }
152. }
153. }
155. statusBar: Rectangle {
156. color: "lightgray";
157. implicitHeight: 30;
158. width: parent.width;
159. property alias text: status.text;
160. Text {
161. id: status;
162. anchors.fill: parent;
163. anchors.margins: 4;
164. font.pointSize: 12;
165. }
166. }
168. Item {
169. id: centralView;
170. anchors.fill: parent;
171. visible: true;
172. property var current: null;
173. BusyIndicator {
174. id: busy;
175. anchors.centerIn: parent;
176. running: false;
177. z: 3;
178. }
179. Image {
180. id: imageViewer;
181. anchors.fill: parent;
182. visible: false;
183. asynchronous: true;
184. fillMode: Image.PreserveAspectFit;
185. onStatusChanged: {
186. if (status === Image.Loading) {
187. centralView.busy.running = true;
188. }
189. else if(status === Image.Ready){
190. centralView.busy.running = false;
191. }
192. else if(status === Image.Error){
193. centralView.busy.running = false;
194. centralView.statusBar.text = "图片无法显示";
195. }
196. }
197. }
199. TextArea {
200. id: textView;
201. anchors.fill: parent;
202. readOnly: true;
203. visible: false;
204. wrapMode: TextEdit.WordWrap;
205. font.pointSize: 12;
206. style: TextAreaStyle{
207. backgroundColor: root.textBackgroundColor;
208. textColor: root.textColor;
209. selectionColor: "steelblue";
210. selectedTextColor: "#a00000";
211. }
213. property var xmlhttp: null;
214. function onReadyStateChanged(){
215. if(xmlhttp.readyState == 4){
216. text = xmlhttp.responseText;
217. xmlhttp.abort();
218. }
219. }
221. function loadText(fileUrl){
222. if(xmlhttp == null){
223. xmlhttp = new XMLHttpRequest();
224. xmlhttp.onreadystatechange = onReadyStateChanged;
225. }
226. if(xmlhttp.readyState == 0){
227. xmlhttp.open("GET", fileUrl);
228. xmlhttp.send(null);
229. }
230. }
231. }
233. VideoOutput {
234. id: videoOutput;
235. anchors.fill: parent;
236. visible: false;
237. source: player;
238. onVisibleChanged: {
239. playerState.visible = visible;
240. }
241. MouseArea {
242. anchors.fill: parent;
243. onClicked: {
244. switch(player.playbackState){
245. case MediaPlayer.PausedState:
246. case MediaPlayer.StoppedState:
247. player.play();
248. break;
249. case MediaPlayer.PlayingState:
250. player.pause();
251. break;
252. }
253. }
254. }
255. }
257. Rectangle {
258. id: playerState;
259. color: "gray";
260. radius: 16;
261. opacity: 0.8;
262. visible: false;
263. z: 2;
264. implicitHeight: 80;
265. implicitWidth: 200;
266. anchors.horizontalCenter: parent.horizontalCenter;
267. anchors.bottom: parent.bottom;
268. anchors.bottomMargin: 16;
269. Column {
270. anchors.fill: parent;
271. anchors.leftMargin: 12;
272. anchors.rightMargin: 12;
273. anchors.topMargin: 6;
274. anchors.bottomMargin: 6;
275. spacing: 4;
276. Text {
277. id: state;
278. font.pointSize: 14;
279. color: "blue";
280. }
281. Text {
282. id: progress;
283. font.pointSize: 12;
284. color: "white";
285. }
286. }
287. }
289. MediaPlayer {
290. id: player;
292. property var utilDate: new Date();
293. function msecs2String(msecs){
294. utilDate.setTime(msecs);
295. return Qt.formatTime(utilDate, "mm:ss");
296. }
297. property var sDuration;
299. onPositionChanged: {
300. progress.text = msecs2String(position) + sDuration;
301. }
302. onDurationChanged: {
303. sDuration = " / " + msecs2String(duration);
304. }
305. onPlaybackStateChanged: {
306. switch(playbackState){
307. case MediaPlayer.PlayingState:
308. state.text = "播放中";
309. break;
310. case MediaPlayer.PausedState:
311. state.text = "已暂停";
312. break;
313. case MediaPlayer.StoppedState:
314. state.text = "停止";
315. break;
316. }
317. }
318. onStatusChanged: {
319. switch(status){
320. case MediaPlayer.Loading:
321. case MediaPlayer.Buffering:
322. busy.running = true;
323. break;
324. case MediaPlayer.InvalidMedia:
325. root.statusBar.text = "无法播放";
326. case MediaPlayer.Buffered:
327. case MediaPlayer.Loaded:
328. busy.running = false;
329. break;
330. }
331. }
332. }
333. }

336. function processFile(fileUrl, ext){
337. var i = 0;
338. for(; i < fileDialog.nameFilters.length; i++){
339. if(fileDialog.nameFilters[i].search(ext) != -1) break;
340. }
341. switch(i){
342. case 0:
343. //text file
344. if(centralView.current != textView){
345. if(centralView.current != null){
346. centralView.current.visible = false;
347. }
348. textView.visible = true;
349. centralView.current = textView;
350. }
351. textView.loadText(fileUrl);
352. break;
353. case 1:
354. if(centralView.current != imageViewer){
355. if(centralView.current != null){
356. centralView.current.visible = false;
357. }
358. imageViewer.visible = true;
359. centralView.current = imageViewer;
360. }
361. imageViewer.source = fileUrl;
362. break;
363. case 2:
364. case 3:
365. if(centralView.current != videoOutput){
366. if(centralView.current != null){
367. centralView.current.visible = false;
368. }
369. videoOutput.visible = true;
370. centralView.current = videoOutput;
371. }
372. player.source = fileUrl;
373. player.play();
374. break;
375. default:
376. statusBar.text = "抱歉，处理不了";
377. break;
378. }
379. }
381. function showAbout(){
382. if(aboutDlg == null){
383. aboutDlg = Qt.createQmlObject(
384. 'import QtQuick 2.2;import QtQuick.Dialogs 1.1;MessageDialog{icon: StandardIcon.Information;title: "关于";\ntext: "仅仅是个示例撒";\nstandardButtons:StandardButton.Ok;}'
385. , root, "aboutDlg");
386. aboutDlg.accepted.connect(onAboutDlgClosed);
387. aboutDlg.rejected.connect(onAboutDlgClosed);
388. aboutDlg.visible = true;
389. }
391. }
393. function selectColor(func){
394. if(colorDlg == null){
395. colorDlg = Qt.createQmlObject(
396. 'import QtQuick 2.2;import QtQuick.Dialogs 1.1;ColorDialog{}',
397. root, "colorDlg");
398. colorDlg.accepted.connect(func);
399. colorDlg.accepted.connect(onColorDlgClosed);
400. colorDlg.rejected.connect(onColorDlgClosed);
401. colorDlg.visible = true;
402. }
403. }
405. function onAboutDlgClosed(){
406. aboutDlg.destroy();
407. aboutDlg = null;
408. }
410. function onColorDlgClosed(){
411. colorDlg.destroy();
412. colorDlg = null;
413. }
415. function onTextColorSelected(){
416. root.textColor = colorDlg.color;
417. }
419. function onTextBackgroundColorSelected(){
420. root.textBackgroundColor = colorDlg.color;
421. }
423. FileDialog {
424. id: fileDialog;
425. title: qsTr("Please choose an image file");
426. nameFilters: [
427. "Text Files (\*.txt \*.ini \*.log \*.c \*.h \*.java \*.cpp \*.html \*.xml)",
428. "Image Files (\*.jpg \*.png \*.gif \*.bmp \*.ico)",
429. "Video Files (\*.ts \*.mp4 \*.avi \*.flv \*.mkv \*.3gp)",
430. "Audio Files (\*.mp3 \*.ogg \*.wav \*.wma \*.ape \*.ra)",
431. "\*.\*"
432. ];
433. onAccepted: {
434. var filepath = new String(fileUrl);
435. //remove file:///
436. if(Qt.platform.os == "windows"){
437. root.statusBar.text = filepath.slice(8);
438. }else{
439. root.statusBar.text = filepath.slice(7);
440. }
441. var dot = filepath.lastIndexOf(".");
442. var sep = filepath.lastIndexOf("/");
443. if(dot > sep){
444. var ext = filepath.substring(dot);
445. root.processFile(fileUrl, ext.toLowerCase());
446. }else{
447. root.statusBar.text = "Not Supported!";
448. }
449. }
450. }
451. }

import QtQuick 2.2

import QtQuick.Window 2.1

import QtQuick.Controls 1.2

import QtQuick.Controls.Styles 1.2

import QtQuick.Layouts 1.1

import QtQuick.Dialogs 1.1

import QtMultimedia 5.0

ApplicationWindow {

visible: true

width: 480

height: 360;

color: "black";

title: "文件查看器";

id: root;

property var aboutDlg: null;

property var colorDlg: null;

property color textColor: "green";

property color textBackgroundColor: "black";

menuBar: MenuBar{

Menu {

title: "文件";

MenuItem{

iconSource: "res/txtFile.png";

action: Action{

id: textAction;

iconSource: "res/txtFile.png";

text: "文本文件";

onTriggered: {

fileDialog.selectedNameFilter = fileDialog.nameFilters[0];

fileDialog.open();

}

tooltip: "打开txt等文本文件";

}

}

MenuItem{

action: Action {

id: imageAction;

text: "图片";

iconSource: "res/imageFile.png";

onTriggered: {

fileDialog.selectedNameFilter = fileDialog.nameFilters[1];

fileDialog.open();

}

tooltip: "打开jpg等格式的图片";

}

}

MenuItem{

action: Action {

id: videoAction;

iconSource: "res/videoFile.png";

text: "视频";

onTriggered: {

fileDialog.selectedNameFilter = fileDialog.nameFilters[2];

fileDialog.open();

}

tooltip: "打开TS、MKV、MP4等格式的文件";

}

}

MenuItem{

action: Action {

id: audioAction;

iconSource: "res/audioFile.png";

text: "音乐";

onTriggered: {

fileDialog.selectedNameFilter = fileDialog.nameFilters[3];

fileDialog.open();

}

tooltip: "打开mp3、wma等格式的文件";

}

}

MenuItem{

text: "退出";

onTriggered: Qt.quit();

}

}

Menu {

title: "设置";

MenuItem {

action: Action {

id: textColorAction;

iconSource: "res/ic\_textcolor.png";

text: "文字颜色";

onTriggered: root.selectColor(root.onTextColorSelected);

}

}

MenuItem {

action: Action{

id: backgroundColorAction;

iconSource: "res/ic\_bkgndcolor.png";

text: "文字背景色";

onTriggered: root.selectColor(root.onTextBackgroundColorSelected);

}

}

MenuItem {

action: Action{

id: fontSizeAddAction;

iconSource: "res/ic\_fontsize2.png";

text: "增大字体";

onTriggered: textView.font.pointSize += 1;

}

}

MenuItem {

action: Action{

id: fontSizeMinusAction;

iconSource: "res/ic\_fontsize1.png";

text: "减小字体";

onTriggered: textView.font.pointSize -= 1;

}

}

}

Menu {

title: "帮助";

MenuItem{

text: "关于";

onTriggered: root.showAbout();

}

MenuItem{

text: "访问作者博客";

onTriggered: Qt.openUrlExternally("http://blog.csdn.net/foruok");

}

}

}

toolBar: ToolBar{

RowLayout {

ToolButton{

action: textAction;

}

ToolButton{

action: imageAction;

}

ToolButton{

action: videoAction;

}

ToolButton{

action: audioAction;

}

ToolButton{

action: textColorAction;

}

ToolButton {

action: backgroundColorAction;

}

ToolButton {

action: fontSizeAddAction;

}

ToolButton {

action: fontSizeMinusAction;

}

}

}

statusBar: Rectangle {

color: "lightgray";

implicitHeight: 30;

width: parent.width;

property alias text: status.text;

Text {

id: status;

anchors.fill: parent;

anchors.margins: 4;

font.pointSize: 12;

}

}

Item {

id: centralView;

anchors.fill: parent;

visible: true;

property var current: null;

BusyIndicator {

id: busy;

anchors.centerIn: parent;

running: false;

z: 3;

}

Image {

id: imageViewer;

anchors.fill: parent;

visible: false;

asynchronous: true;

fillMode: Image.PreserveAspectFit;

onStatusChanged: {

if (status === Image.Loading) {

centralView.busy.running = true;

}

else if(status === Image.Ready){

centralView.busy.running = false;

}

else if(status === Image.Error){

centralView.busy.running = false;

centralView.statusBar.text = "图片无法显示";

}

}

}

TextArea {

id: textView;

anchors.fill: parent;

readOnly: true;

visible: false;

wrapMode: TextEdit.WordWrap;

font.pointSize: 12;

style: TextAreaStyle{

backgroundColor: root.textBackgroundColor;

textColor: root.textColor;

selectionColor: "steelblue";

selectedTextColor: "#a00000";

}

property var xmlhttp: null;

function onReadyStateChanged(){

if(xmlhttp.readyState == 4){

text = xmlhttp.responseText;

xmlhttp.abort();

}

}

function loadText(fileUrl){

if(xmlhttp == null){

xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = onReadyStateChanged;

}

if(xmlhttp.readyState == 0){

xmlhttp.open("GET", fileUrl);

xmlhttp.send(null);

}

}

}

VideoOutput {

id: videoOutput;

anchors.fill: parent;

visible: false;

source: player;

onVisibleChanged: {

playerState.visible = visible;

}

MouseArea {

anchors.fill: parent;

onClicked: {

switch(player.playbackState){

case MediaPlayer.PausedState:

case MediaPlayer.StoppedState:

player.play();

break;

case MediaPlayer.PlayingState:

player.pause();

break;

}

}

}

}

Rectangle {

id: playerState;

color: "gray";

radius: 16;

opacity: 0.8;

visible: false;

z: 2;

implicitHeight: 80;

implicitWidth: 200;

anchors.horizontalCenter: parent.horizontalCenter;

anchors.bottom: parent.bottom;

anchors.bottomMargin: 16;

Column {

anchors.fill: parent;

anchors.leftMargin: 12;

anchors.rightMargin: 12;

anchors.topMargin: 6;

anchors.bottomMargin: 6;

spacing: 4;

Text {

id: state;

font.pointSize: 14;

color: "blue";

}

Text {

id: progress;

font.pointSize: 12;

color: "white";

}

}

}

MediaPlayer {

id: player;

property var utilDate: new Date();

function msecs2String(msecs){

utilDate.setTime(msecs);

return Qt.formatTime(utilDate, "mm:ss");

}

property var sDuration;

onPositionChanged: {

progress.text = msecs2String(position) + sDuration;

}

onDurationChanged: {

sDuration = " / " + msecs2String(duration);

}

onPlaybackStateChanged: {

switch(playbackState){

case MediaPlayer.PlayingState:

state.text = "播放中";

break;

case MediaPlayer.PausedState:

state.text = "已暂停";

break;

case MediaPlayer.StoppedState:

state.text = "停止";

break;

}

}

onStatusChanged: {

switch(status){

case MediaPlayer.Loading:

case MediaPlayer.Buffering:

busy.running = true;

break;

case MediaPlayer.InvalidMedia:

root.statusBar.text = "无法播放";

case MediaPlayer.Buffered:

case MediaPlayer.Loaded:

busy.running = false;

break;

}

}

}

}

function processFile(fileUrl, ext){

var i = 0;

for(; i < fileDialog.nameFilters.length; i++){

if(fileDialog.nameFilters[i].search(ext) != -1) break;

}

switch(i){

case 0:

//text file

if(centralView.current != textView){

if(centralView.current != null){

centralView.current.visible = false;

}

textView.visible = true;

centralView.current = textView;

}

textView.loadText(fileUrl);

break;

case 1:

if(centralView.current != imageViewer){

if(centralView.current != null){

centralView.current.visible = false;

}

imageViewer.visible = true;

centralView.current = imageViewer;

}

imageViewer.source = fileUrl;

break;

case 2:

case 3:

if(centralView.current != videoOutput){

if(centralView.current != null){

centralView.current.visible = false;

}

videoOutput.visible = true;

centralView.current = videoOutput;

}

player.source = fileUrl;

player.play();

break;

default:

statusBar.text = "抱歉，处理不了";

break;

}

}

function showAbout(){

if(aboutDlg == null){

aboutDlg = Qt.createQmlObject(

'import QtQuick 2.2;import QtQuick.Dialogs 1.1;MessageDialog{icon: StandardIcon.Information;title: "关于";\ntext: "仅仅是个示例撒";\nstandardButtons:StandardButton.Ok;}'

, root, "aboutDlg");

aboutDlg.accepted.connect(onAboutDlgClosed);

aboutDlg.rejected.connect(onAboutDlgClosed);

aboutDlg.visible = true;

}

}

function selectColor(func){

if(colorDlg == null){

colorDlg = Qt.createQmlObject(

'import QtQuick 2.2;import QtQuick.Dialogs 1.1;ColorDialog{}',

root, "colorDlg");

colorDlg.accepted.connect(func);

colorDlg.accepted.connect(onColorDlgClosed);

colorDlg.rejected.connect(onColorDlgClosed);

colorDlg.visible = true;

}

}

function onAboutDlgClosed(){

aboutDlg.destroy();

aboutDlg = null;

}

function onColorDlgClosed(){

colorDlg.destroy();

colorDlg = null;

}

function onTextColorSelected(){

root.textColor = colorDlg.color;

}

function onTextBackgroundColorSelected(){

root.textBackgroundColor = colorDlg.color;

}

FileDialog {

id: fileDialog;

title: qsTr("Please choose an image file");

nameFilters: [

"Text Files (\*.txt \*.ini \*.log \*.c \*.h \*.java \*.cpp \*.html \*.xml)",

"Image Files (\*.jpg \*.png \*.gif \*.bmp \*.ico)",

"Video Files (\*.ts \*.mp4 \*.avi \*.flv \*.mkv \*.3gp)",

"Audio Files (\*.mp3 \*.ogg \*.wav \*.wma \*.ape \*.ra)",

"\*.\*"

];

onAccepted: {

var filepath = new String(fileUrl);

//remove file:///

if(Qt.platform.os == "windows"){

root.statusBar.text = filepath.slice(8);

}else{

root.statusBar.text = filepath.slice(7);

}

var dot = filepath.lastIndexOf(".");

var sep = filepath.lastIndexOf("/");

if(dot > sep){

var ext = filepath.substring(dot);

root.processFile(fileUrl, ext.toLowerCase());

}else{

root.statusBar.text = "Not Supported!";

}

}

}

}

**C++代码**

    其实，我只对模板生成的C++代码改动了三行，一行include，一行QApplication，一行设置应用图标。main.cpp如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. #include <QApplication>
2. #include <QQmlApplicationEngine>
3. #include <QIcon>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QApplication app(argc, argv);
8. app.setWindowIcon(QIcon(":/res/eye.png"));
10. QQmlApplicationEngine engine;
11. engine.load(QUrl(QStringLiteral("qrc:///main.qml")));
13. return app.exec();
14. }

#include <QApplication>

#include <QQmlApplicationEngine>

#include <QIcon>

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

app.setWindowIcon(QIcon(":/res/eye.png"));

QQmlApplicationEngine engine;

engine.load(QUrl(QStringLiteral("qrc:///main.qml")));

return app.exec();

}

**PRO文件**

    有人喊，兄弟，这也要贴！你凑字数呢……你管，就放这里了：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39005703)

1. TEMPLATE = app
3. QT += qml quick network multimedia widgets
5. SOURCES += main.cpp
7. RESOURCES += qml.qrc
9. # Additional import path used to resolve QML modules in Qt Creator's code model
10. QML\_IMPORT\_PATH =
12. # Default rules for deployment.
13. include(deployment.pri)
15. HEADERS +=

TEMPLATE = app

QT += qml quick network multimedia widgets

SOURCES += main.cpp

RESOURCES += qml.qrc

# Additional import path used to resolve QML modules in Qt Creator's code model

QML\_IMPORT\_PATH =

# Default rules for deployment.

include(deployment.pri)

HEADERS +=

    版权所有foruok，转载请注明出处：<http://blog.csdn.net/foruok>。

    好啦，彪悍的人生不需要解释，都扒光了，你自己仔细看吧。